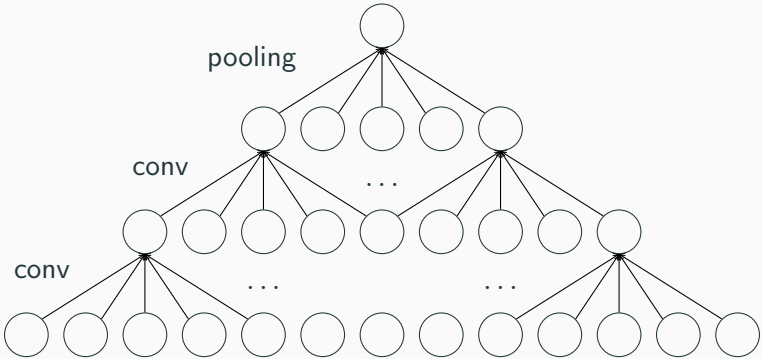
Convolutional Networks 2: Training, deep convolutional networks

Hakan Bilen

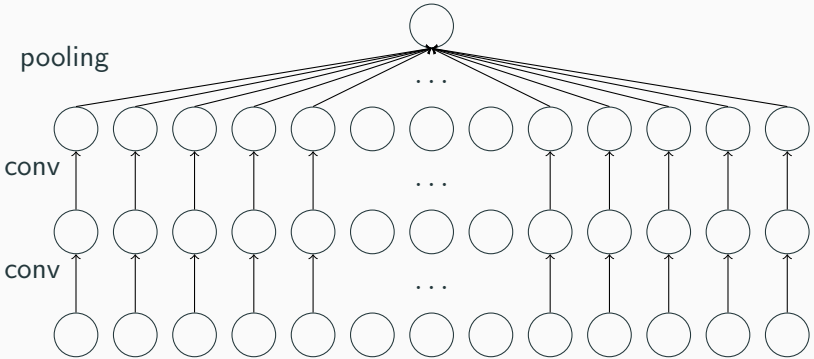
Machine Learning Practical — MLP Lecture 8
29 October / 5 November 2019

Recap: Receptive field (kernel size = 5)

☺ As the depth increases, the local receptive field gets larger.

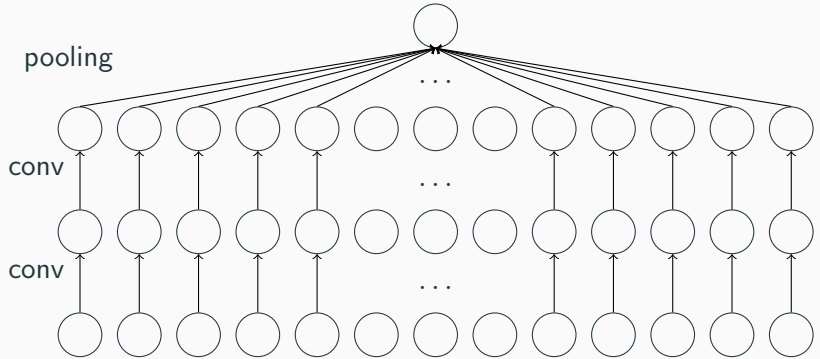


Receptive field (kernel size = 1)



Receptive field (kernel size = 1)

☹ A bad network design: Pooled pixel classifiers!



Increasing receptive field size

- Increasing kernel size: additional weights and computation
- Pooling: loss of resolution

Increasing receptive field size

- Increasing kernel size: additional weights and computation
- Pooling: loss of resolution

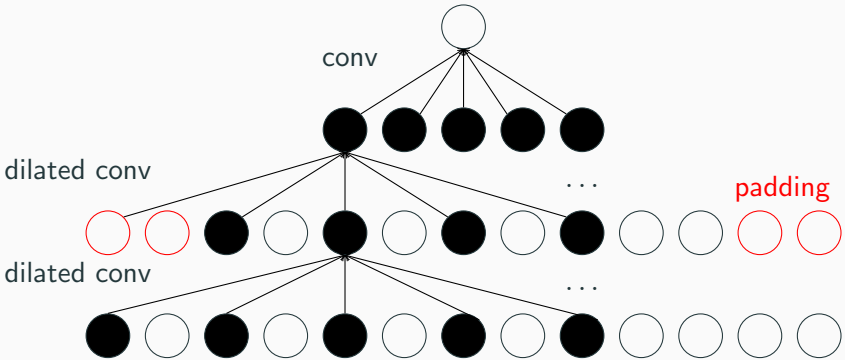
Dilated convolution:

- Enlarges receptive field by **inflating** the kernel by inserting $d - 1$ spaces between the kernel elements
- Results in same number of weights and computations
- Preserves the original resolution with padding

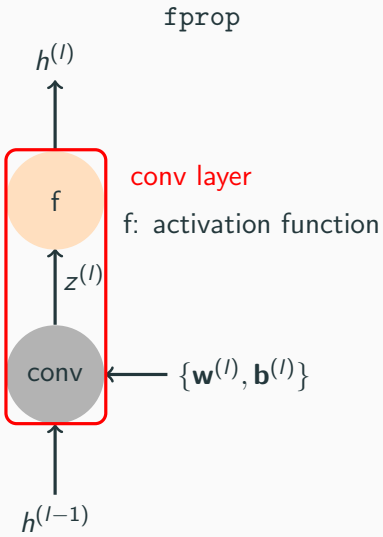
Yu & Koltun, "Multi-scale context aggregation by dilated convolutions", *ICLR*, 2016.

Dilated convolutions ($d = 2$)

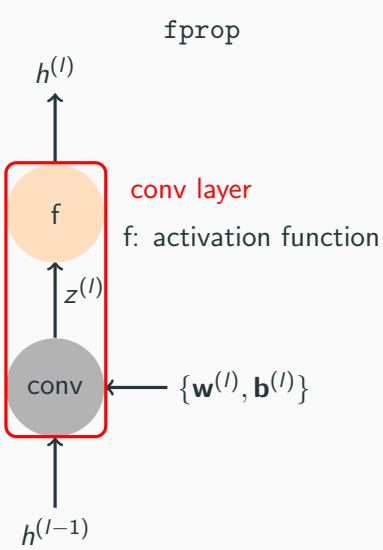
Enlarges receptive field by **inflating** the kernel by inserting $d - 1$ spaces between the kernel elements

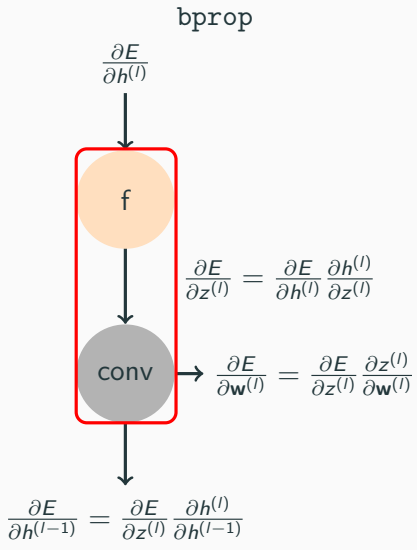


Training Convolutional Networks



Training Convolutional Networks





Example

$$\underbrace{\begin{pmatrix} z_{00}^{(l)} & z_{01}^{(l)} \\ z_{10}^{(l)} & z_{11}^{(l)} \end{pmatrix}}_{z^{(l)}} = \underbrace{\begin{pmatrix} h_{00}^{(l-1)} & h_{01}^{(l-1)} & h_{02}^{(l-1)} \\ h_{10}^{(l-1)} & h_{11}^{(l-1)} & h_{12}^{(l-1)} \\ h_{20}^{(l-1)} & h_{21}^{(l-1)} & h_{22}^{(l-1)} \end{pmatrix}}_{h^{(l-1)}} \otimes \underbrace{\begin{pmatrix} w_{00}^{(l)} & w_{01}^{(l)} \\ w_{10}^{(l)} & w_{11}^{(l)} \end{pmatrix}}_{w^{(l)}} + b$$

$$\begin{aligned}
z_{00}^{(l)} &= w_{00}^{(l)} h_{00}^{(l-1)} + w_{01}^{(l)} h_{01}^{(l-1)} + w_{10}^{(l)} h_{10}^{(l-1)} + w_{11}^{(l)} h_{11}^{(l-1)} + b \\
z_{01}^{(l)} &= w_{00}^{(l)} h_{01}^{(l-1)} + w_{01}^{(l)} h_{02}^{(l-1)} + w_{10}^{(l)} h_{11}^{(l-1)} + w_{11}^{(l)} h_{12}^{(l-1)} + b \\
z_{10}^{(l)} &= w_{00}^{(l)} h_{10}^{(l-1)} + w_{01}^{(l)} h_{11}^{(l-1)} + w_{10}^{(l)} h_{20}^{(l-1)} + w_{11}^{(l)} h_{21}^{(l-1)} + b \\
z_{11}^{(l)} &= w_{00}^{(l)} h_{11}^{(l-1)} + w_{01}^{(l)} h_{12}^{(l-1)} + w_{10}^{(l)} h_{21}^{(l-1)} + w_{11}^{(l)} h_{22}^{(l-1)} + b
\end{aligned}$$

Gradients of E w.r.t $\mathbf{w}^{(l)}$

$$
\begin{aligned}
z_{00}^{(l)} &= w_{00}^{(l)} h_{00}^{(l-1)} + w_{01}^{(l)} h_{01}^{(l-1)} + w_{10}^{(l)} h_{10}^{(l-1)} + w_{11}^{(l)} h_{11}^{(l-1)} + b \\
z_{01}^{(l)} &= w_{00}^{(l)} h_{01}^{(l-1)} + w_{01}^{(l)} h_{02}^{(l-1)} + w_{10}^{(l)} h_{11}^{(l-1)} + w_{11}^{(l)} h_{12}^{(l-1)} + b \\
z_{10}^{(l)} &= w_{00}^{(l)} h_{10}^{(l-1)} + w_{01}^{(l)} h_{11}^{(l-1)} + w_{10}^{(l)} h_{20}^{(l-1)} + w_{11}^{(l)} h_{21}^{(l-1)} + b \\
z_{11}^{(l)} &= w_{00}^{(l)} h_{11}^{(l-1)} + w_{01}^{(l)} h_{12}^{(l-1)} + w_{10}^{(l)} h_{21}^{(l-1)} + w_{11}^{(l)} h_{22}^{(l-1)} + b
\end{aligned}
$$

Let's derive the parameter updates ($\frac{\partial E}{\partial \mathbf{w}^{(l)}}$)

$$
\frac{\partial E}{\partial w_{00}^{(l)}} = \frac{\partial E}{\partial z_{00}^{(l)}} \frac{\partial z_{00}^{(l)}}{\partial w_{00}^{(l)}} + \frac{\partial E}{\partial z_{01}^{(l)}} \frac{\partial z_{01}^{(l)}}{\partial w_{00}^{(l)}} + \frac{\partial E}{\partial z_{10}^{(l)}} \frac{\partial z_{10}^{(l)}}{\partial w_{00}^{(l)}} + \frac{\partial E}{\partial z_{11}^{(l)}} \frac{\partial z_{11}^{(l)}}{\partial w_{00}^{(l)}}
$$

Gradients of E w.r.t $\mathbf{w}^{(l)}$

$$
\begin{aligned}
z_{00}^{(l)} &= w_{00}^{(l)} h_{00}^{(l-1)} + w_{01}^{(l)} h_{01}^{(l-1)} + w_{10}^{(l)} h_{10}^{(l-1)} + w_{11}^{(l)} h_{11}^{(l-1)} + b \\
z_{01}^{(l)} &= w_{00}^{(l)} h_{01}^{(l-1)} + w_{01}^{(l)} h_{02}^{(l-1)} + w_{10}^{(l)} h_{11}^{(l-1)} + w_{11}^{(l)} h_{12}^{(l-1)} + b \\
z_{10}^{(l)} &= w_{00}^{(l)} h_{10}^{(l-1)} + w_{01}^{(l)} h_{11}^{(l-1)} + w_{10}^{(l)} h_{20}^{(l-1)} + w_{11}^{(l)} h_{21}^{(l-1)} + b \\
z_{11}^{(l)} &= w_{00}^{(l)} h_{11}^{(l-1)} + w_{01}^{(l)} h_{12}^{(l-1)} + w_{10}^{(l)} h_{21}^{(l-1)} + w_{11}^{(l)} h_{22}^{(l-1)} + b
\end{aligned}
$$

Let's derive the parameter updates ($\frac{\partial E}{\partial \mathbf{w}^{(l)}}$)

$$
\frac{\partial E}{\partial w_{00}^{(l)}} = \frac{\partial E}{\partial z_{00}^{(l)}} h_{00}^{(l-1)} + \frac{\partial E}{\partial z_{01}^{(l)}} h_{01}^{(l-1)} + \frac{\partial E}{\partial z_{10}^{(l)}} h_{10}^{(l-1)} + \frac{\partial E}{\partial z_{11}^{(l)}} h_{11}^{(l-1)}
$$

Gradients of E w.r.t $\mathbf{w}^{(l)}$

$$\frac{\partial E}{\partial w_{00}^{(l)}} = \frac{\partial E}{\partial z_{00}^{(l)}} h_{00}^{(l-1)} + \frac{\partial E}{\partial z_{01}^{(l)}} h_{01}^{(l-1)} + \frac{\partial E}{\partial z_{10}^{(l)}} h_{10}^{(l-1)} + \frac{\partial E}{\partial z_{11}^{(l)}} h_{11}^{(l-1)}$$
$$\frac{\partial E}{\partial w_{01}^{(l)}} = \frac{\partial E}{\partial z_{00}^{(l)}} h_{01}^{(l-1)} + \frac{\partial E}{\partial z_{01}^{(l)}} h_{02}^{(l-1)} + \frac{\partial E}{\partial z_{10}^{(l)}} h_{11}^{(l-1)} + \frac{\partial E}{\partial z_{11}^{(l)}} h_{12}^{(l-1)}$$
$$\frac{\partial E}{\partial w_{10}^{(l)}} = \frac{\partial E}{\partial z_{00}^{(l)}} h_{10}^{(l-1)} + \frac{\partial E}{\partial z_{01}^{(l)}} h_{11}^{(l-1)} + \frac{\partial E}{\partial z_{10}^{(l)}} h_{20}^{(l-1)} + \frac{\partial E}{\partial z_{11}^{(l)}} h_{21}^{(l-1)}$$
$$\frac{\partial E}{\partial w_{11}^{(l)}} = \frac{\partial E}{\partial z_{00}^{(l)}} h_{11}^{(l-1)} + \frac{\partial E}{\partial z_{01}^{(l)}} h_{12}^{(l-1)} + \frac{\partial E}{\partial z_{10}^{(l)}} h_{21}^{(l-1)} + \frac{\partial E}{\partial z_{11}^{(l)}} h_{22}^{(l-1)}$$

Assume that $z^{(l)}$ is $m \times n$ dimensional

$$\frac{\partial E}{\partial w_{p,q}^{(l)}} = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \frac{\partial E}{\partial z_{i,j}^{(l)}} h_{p+i,q+j}^{(l-1)}$$

Gradients of E w.r.t $\mathbf{w}^{(l)}$

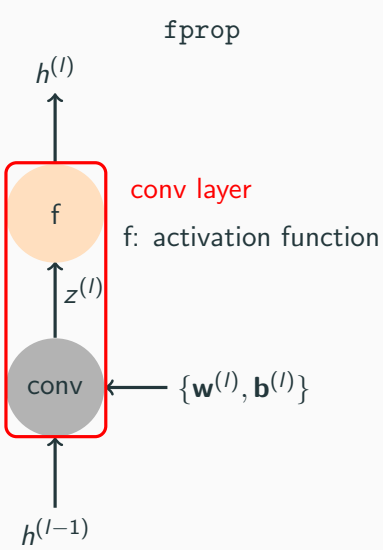
$$\frac{\partial E}{\partial w_{p,q}^{(l)}} = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} h_{p+i,q+j}^{(l-1)} \frac{\partial E}{\partial z_{i,j}^{(l)}}$$

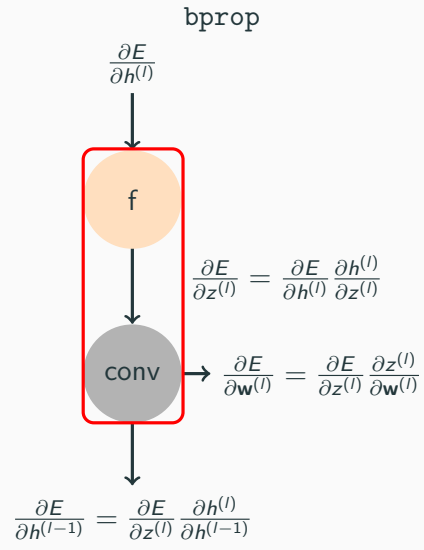
Gradients of E w.r.t $\mathbf{w}^{(l)}$ can be computed as cross-correlation between the previous feature map ($h^{(l-1)}$) and gradients of E w.r.t $z^{(l)}$

.

$$\underbrace{\begin{pmatrix} \frac{\partial E}{\partial w_{00}^{(l)}} & \frac{\partial E}{\partial w_{01}^{(l)}} \\ \frac{\partial E}{\partial w_{10}^{(l)}} & \frac{\partial E}{\partial w_{11}^{(l)}} \end{pmatrix}}_{\frac{\partial E}{\partial w^{(l)}}} = \underbrace{\begin{pmatrix} h_{00}^{(l-1)} & h_{01}^{(l-1)} & h_{02}^{(l-1)} \\ h_{10}^{(l-1)} & h_{11}^{(l-1)} & h_{12}^{(l-1)} \\ h_{20}^{(l-1)} & h_{21}^{(l-1)} & h_{22}^{(l-1)} \end{pmatrix}}_{h^{(l-1)}} \otimes \underbrace{\begin{pmatrix} \frac{\partial E}{\partial z_{00}^{(l)}} & \frac{\partial E}{\partial z_{01}^{(l)}} \\ \frac{\partial E}{\partial z_{10}^{(l)}} & \frac{\partial E}{\partial z_{11}^{(l)}} \end{pmatrix}}_{\frac{\partial E}{\partial z^{(l)}}}$$

Gradients of E w.r.t $z^{(l-1)}$





Gradients of E w.r.t $z^{(l-1)}$

$$
\begin{aligned}
z_{00}^{(l)} &= w_{00}^{(l)} h_{00}^{(l-1)} + w_{01}^{(l)} h_{01}^{(l-1)} + w_{10}^{(l)} h_{10}^{(l-1)} + w_{11}^{(l)} h_{11}^{(l-1)} + b \\
z_{01}^{(l)} &= w_{00}^{(l)} h_{01}^{(l-1)} + w_{01}^{(l)} h_{02}^{(l-1)} + w_{10}^{(l)} h_{11}^{(l-1)} + w_{11}^{(l)} h_{12}^{(l-1)} + b \\
z_{10}^{(l)} &= w_{00}^{(l)} h_{10}^{(l-1)} + w_{01}^{(l)} h_{11}^{(l-1)} + w_{10}^{(l)} h_{20}^{(l-1)} + w_{11}^{(l)} h_{21}^{(l-1)} + b \\
z_{11}^{(l)} &= w_{00}^{(l)} h_{11}^{(l-1)} + w_{01}^{(l)} h_{12}^{(l-1)} + w_{10}^{(l)} h_{21}^{(l-1)} + w_{11}^{(l)} h_{22}^{(l-1)} + b
\end{aligned}
$$

Let's derive the gradients of error function (E) w.r.t previous feature map ($h^{(l-1)}$)

$$
\frac{\partial E}{\partial h_{00}^{(l-1)}} = \frac{\partial E}{\partial z_{00}^{(l)}} \frac{\partial z_{00}^{(l)}}{\partial h_{00}^{(l-1)}} + \frac{\partial E}{\partial z_{01}^{(l)}} \frac{\partial z_{01}^{(l)}}{\partial h_{00}^{(l-1)}} + \frac{\partial E}{\partial z_{10}^{(l)}} \frac{\partial z_{10}^{(l)}}{\partial h_{00}^{(l-1)}} + \frac{\partial E}{\partial z_{11}^{(l)}} \frac{\partial z_{11}^{(l)}}{\partial h_{00}^{(l-1)}}
$$

Gradients of E w.r.t $z^{(l-1)}$

$$
\begin{aligned}
z_{00}^{(l)} &= w_{00}^{(l)} h_{00}^{(l-1)} + w_{01}^{(l)} h_{01}^{(l-1)} + w_{10}^{(l)} h_{10}^{(l-1)} + w_{11}^{(l)} h_{11}^{(l-1)} + b \\
z_{01}^{(l)} &= w_{00}^{(l)} h_{01}^{(l-1)} + w_{01}^{(l)} h_{02}^{(l-1)} + w_{10}^{(l)} h_{11}^{(l-1)} + w_{11}^{(l)} h_{12}^{(l-1)} + b \\
z_{10}^{(l)} &= w_{00}^{(l)} h_{10}^{(l-1)} + w_{01}^{(l)} h_{11}^{(l-1)} + w_{10}^{(l)} h_{20}^{(l-1)} + w_{11}^{(l)} h_{21}^{(l-1)} + b \\
z_{11}^{(l)} &= w_{00}^{(l)} h_{11}^{(l-1)} + w_{01}^{(l)} h_{12}^{(l-1)} + w_{10}^{(l)} h_{21}^{(l-1)} + w_{11}^{(l)} h_{22}^{(l-1)} + b
\end{aligned}
$$

Let's derive the gradients of error function (E) w.r.t previous feature map ($h^{(l-1)}$)

$$
\frac{\partial E}{\partial h_{00}^{(l-1)}} = \frac{\partial E}{\partial z_{00}^{(l)}} w_{00}^{(l)}
$$

Gradients of E w.r.t $z^{(l-1)}$

$$
\begin{aligned}
\frac{\partial E}{\partial h_{00}^{(l-1)}} &= \frac{\partial E}{\partial z_{00}^{(l)}} w_{00}^{(l)} \\
\frac{\partial E}{\partial h_{01}^{(l-1)}} &= \frac{\partial E}{\partial z_{00}^{(l)}} w_{01}^{(l)} + \frac{\partial E}{\partial z_{01}^{(l)}} w_{00}^{(l)} \\
\frac{\partial E}{\partial h_{02}^{(l-1)}} &= \frac{\partial E}{\partial z_{01}^{(l)}} w_{01}^{(l)} \\
&\vdots \\
\frac{\partial E}{\partial h_{11}^{(l-1)}} &= \frac{\partial E}{\partial z_{00}^{(l)}} w_{11}^{(l)} + \frac{\partial E}{\partial z_{01}^{(l)}} w_{10}^{(l)} + \frac{\partial E}{\partial z_{10}^{(l)}} w_{01}^{(l)} + \frac{\partial E}{\partial z_{11}^{(l)}} w_{00}^{(l)} \\
&\vdots \\
\frac{\partial E}{\partial h_{22}^{(l-1)}} &= \frac{\partial E}{\partial z_{11}^{(l)}} w_{11}^{(l)}
\end{aligned}
$$

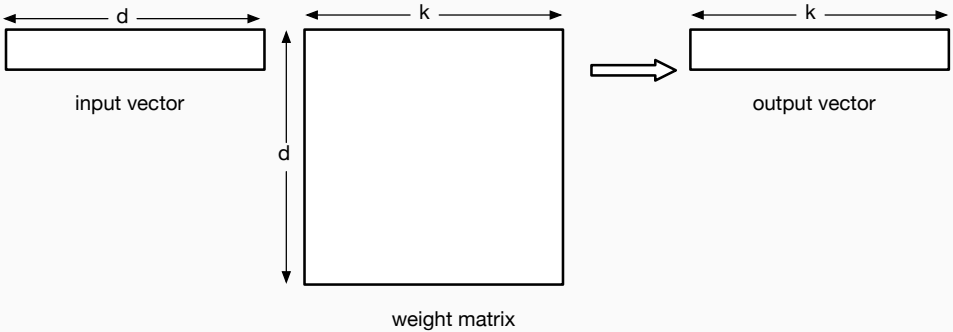
Gradients of E w.r.t $h^{(l-1)}$

Gradients of E w.r.t the previous feature map $h^{(l-1)}$ can be computed as cross-correlation between the "padded" gradients of E w.r.t $z^{(l)}$ and 180^0 rotated conv. kernel $\mathbf{w}^{(l)}$

$$\underbrace{\begin{pmatrix} \frac{\partial E}{\partial h_{00}^{(l-1)}} & \frac{\partial E}{\partial h_{01}^{(l-1)}} & \frac{\partial E}{\partial h_{02}^{(l-1)}} \\ \frac{\partial E}{\partial h_{10}^{(l-1)}} & \frac{\partial E}{\partial h_{11}^{(l-1)}} & \frac{\partial E}{\partial h_{12}^{(l-1)}} \\ \frac{\partial E}{\partial h_{20}^{(l-1)}} & \frac{\partial E}{\partial h_{21}^{(l-1)}} & \frac{\partial E}{\partial h_{22}^{(l-1)}} \end{pmatrix}}_{\frac{\partial E}{\partial h^{(l-1)}}} = \underbrace{\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{\partial E}{\partial z_{00}^{(l)}} & \frac{\partial E}{\partial z_{01}^{(l)}} & 0 \\ 0 & \frac{\partial E}{\partial z_{10}^{(l)}} & \frac{\partial E}{\partial z_{11}^{(l)}} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}}_{\text{Padded } \frac{\partial E}{\partial z^{(l)}}} \otimes \underbrace{\begin{pmatrix} w_{11}^{(l)} & w_{10}^{(l)} \\ w_{01}^{(l)} & w_{00}^{(l)} \end{pmatrix}}_{\text{Rotated } \mathbf{w}^{(l)}}$$

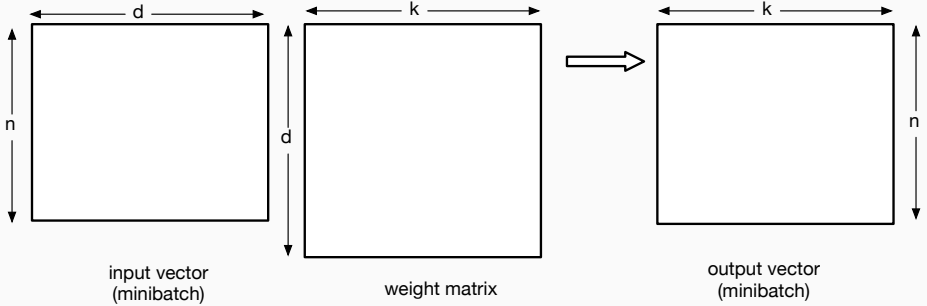
Implementing fully-connected networks

Example at a time:



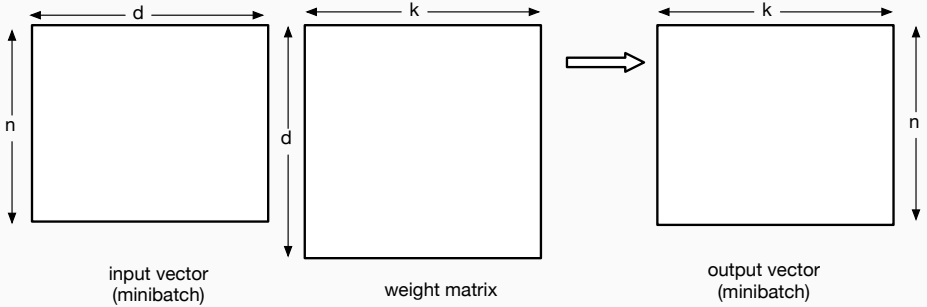
Implementing fully-connected networks

Minibatch:



Implementing fully-connected networks

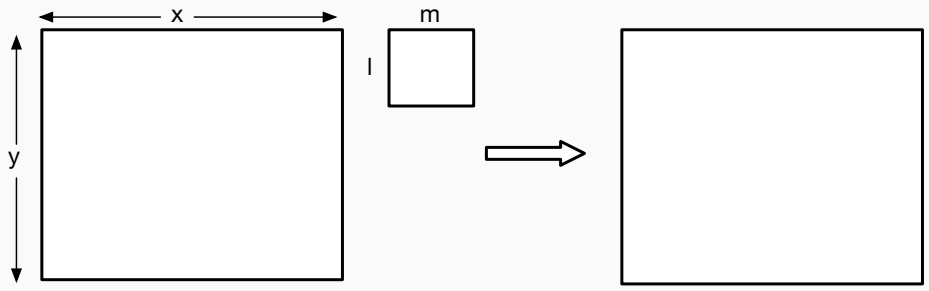
Minibatch:



input dimension x minibatch: Represent each layer as a 2-dimension matrix, where each row corresponds to a training example, and the number of minibatch examples is the number of rows

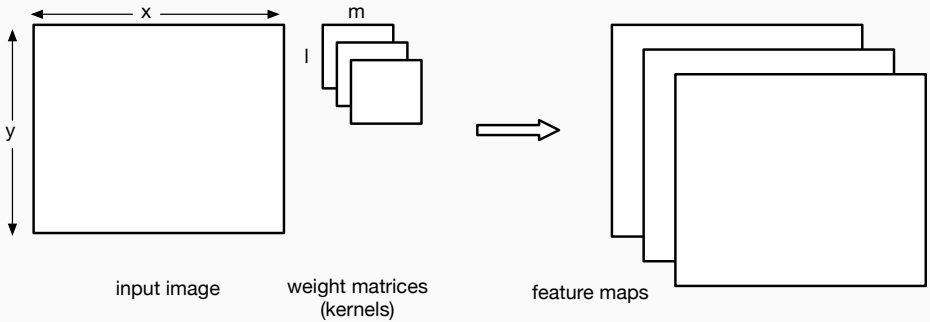
Implementing Convolutional Networks

Example at a time, single input image, single feature map:



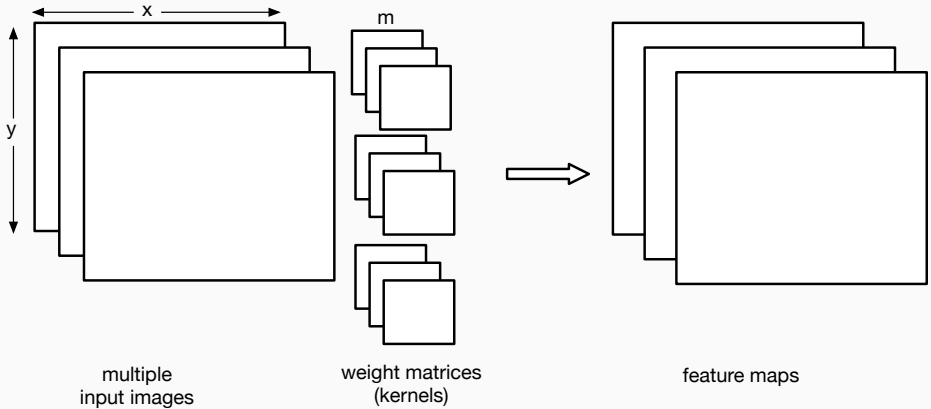
Implementing Convolutional Networks

Example at a time, single input image, multiple feature map:



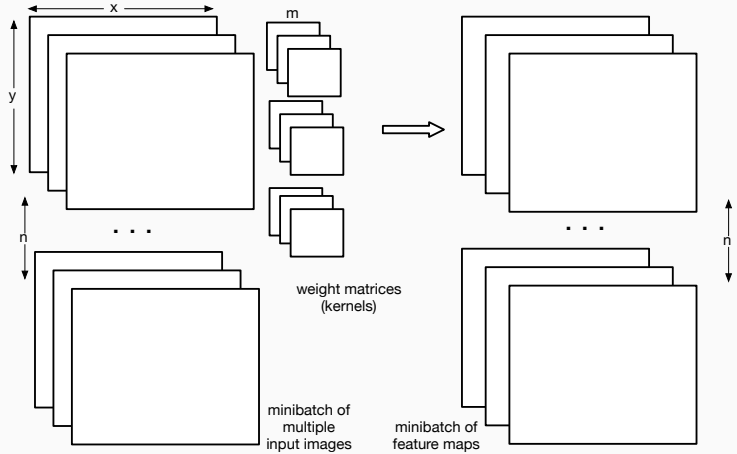
Implementing Convolutional Networks

Example at a time, multiple input images, multiple feature map:



Implementing Convolutional Networks

Minibatch, multiple input images, multiple feature map:



Implementing Convolutional Networks

- Inputs / layer values:
  - Each input image (and convolutional and pooling layer) is 2-dimensions (x,y)
  - If we have multiple feature maps, then that is a third dimension
  - And the minibatch adds a fourth dimension
  - Thus we represent each input (layer values) using a 4-dimension *tensor* (array): (minibatch-size, num-feature-maps, x, y)
- Weight matrices (kernels)
  - Each weight matrix used to scan across an image has 2 spatial dimensions (x,y)
  - If there are multiple feature maps to be computed, then that is a third dimension
  - Multiple input feature maps adds a fourth dimension
  - Thus the weight matrices are also represented using a 4-dimension tensor: (F_{in}, F_{out}, x, y)

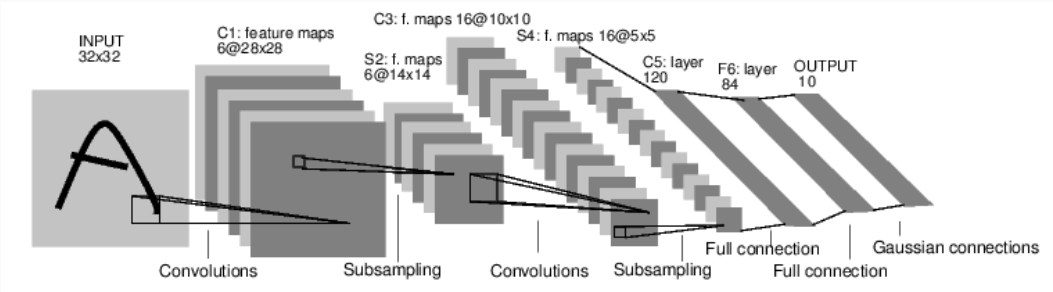
4D tensors in numpy

Both forward and back prop thus involves multiplying 4D tensors. There are various ways to do this:

- Explicitly loop over the dimensions: this results in simpler code, but can be inefficient. Although using cython to compile the loops as C can speed things up
- Serialisation: By replicating input patches and weight matrices, it is possible to convert the required 4D tensor multiplications into a large dot product. Requires careful manipulation of indices!
- Convolutions: use explicit convolution functions for forward and back prop, rotating for the backprop

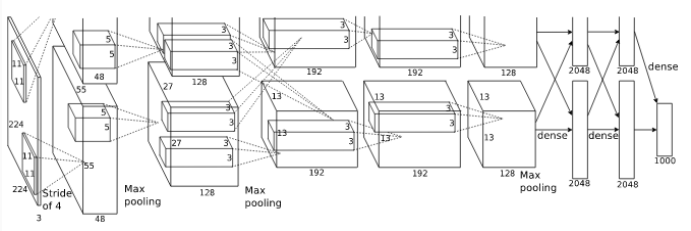
Deep convolutional networks

LeNet5 (LeCun et al, 1997)



- 2 convolutional layers {C1, C3} + non-linearity (tanh)
- 2 average pooling {S2, S4}
- 2 fully connected hidden layer (no weight sharing) {C5, F6}
- Softmax classifier layer

ImageNet Classification ("AlexNet") 2012



Dual GPU architecture

Model	Top-1	Top-5
Sparse coding [2]	*47.1%*	*28.2%*
SIFT + FVs [24]	*45.7%*	*25.7%*
CNN	**37.5%**	**17.0%**

Error rates in the ILSVRC-2012

- Outperformed the that uses hand-crafted features in object classification
- 5 convolutional layers with ReLU + 3 fully connected layers
- 3 max pooling layers

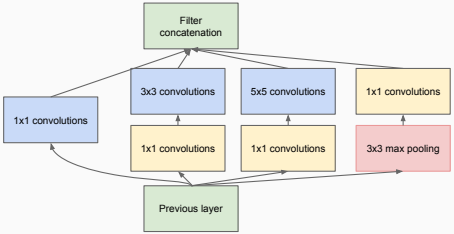
Krizhevsky, Sutskever and Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", NIPS'12. http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

ImageNet Classification ("VGGNet") 2014

- deeper than AlexNet, 13 conv. + 3 fc layers
- 3×3 conv. kernels – very small
- conv. stride 1 – no loss of resolution
- ReLU non-linearity
- vanishing/exploding gradients problem: train shallower networks and gradually add new layers

Simonyan and Zisserman, "Very Deep Convolutional Networks for Large-Scale Visual Recognition", ICLR 2015.

ImageNet Classification ("GoogLeNet") 2014



- objects vary in scale so we should use multiple kernel sizes $(1, 3, 5)$ in parallel
- outputs of the parallel layers are concatenated
- the number of feature maps are reduced with $F_{in} \times F_{in}/2 \times 1 \times 1$ conv. to lower computational load

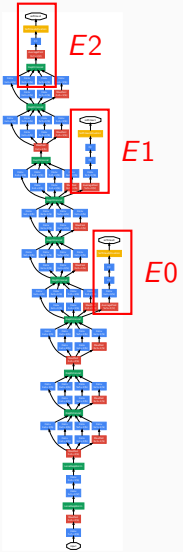
Szegedy et al., "Going deeper with convolutions", CVPR 2015. Image credits

ImageNet Classification ("GoogLeNet") 2014

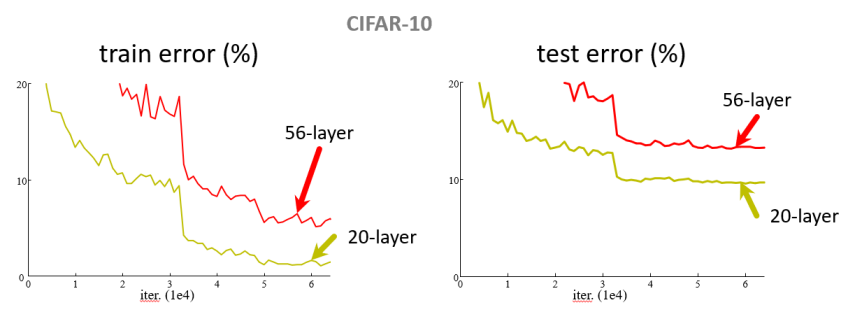
- 22 layer deep (27 including pooling layers)
- Use two "auxiliary classifiers" to middle layers to alleviate the vanishing gradient problem
- Optimizes a weighted sum of "auxiliary errors" ($\alpha_0 = 0.3$ and $\alpha_1 = 0.3$)

$$E = \alpha_0 E_0 + \alpha_1 E_1 + E_2$$

Szegedy et al., "Going deeper with convolutions", CVPR 2015. Image credits



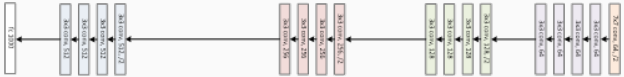
Simply stacking more layers?



Though it can be trained (with the help of Batch Norm),
56-layer net has higher training error and test error than 20-layer net!

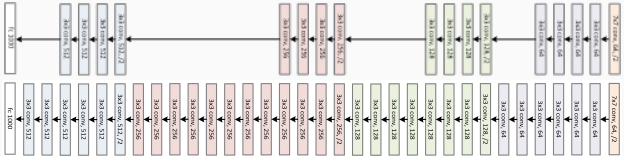
He et al, "Deep Residual Learning for Image Recognition", CVPR-2016.

Deep Residual Learning ("ResNets")



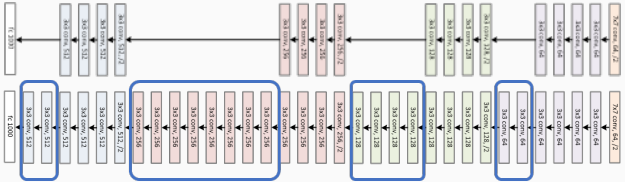
He et al, "Deep Residual Learning for Image Recognition", CVPR-2016.

Deep Residual Learning ("ResNets")



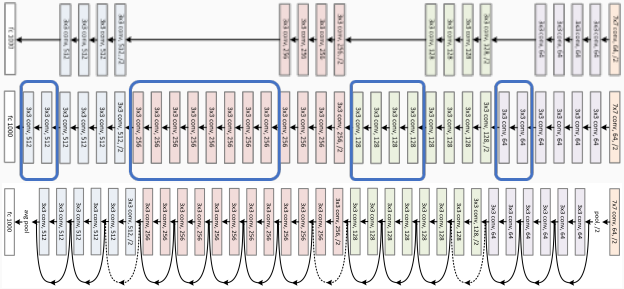
He et al, "Deep Residual Learning for Image Recognition", CVPR-2016.

Deep Residual Learning ("ResNets")



He et al, "Deep Residual Learning for Image Recognition", CVPR-2016.

Deep Residual Learning ("ResNets")



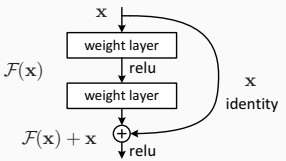


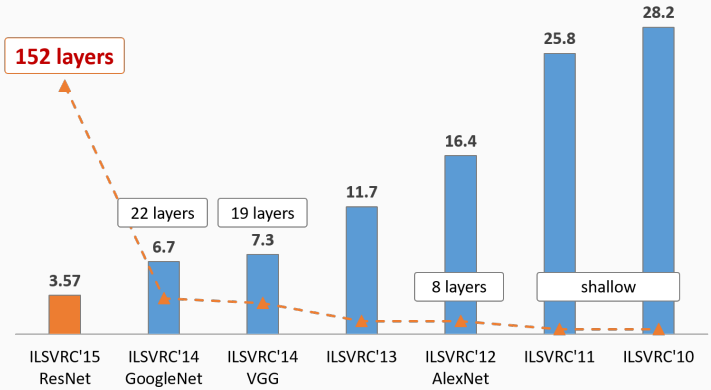
Figure 2. Residual learning: a building block.

A solution by construction:

- original layers: copied from a learned shallower model
- extra layers: set as identity
- at least the same training error

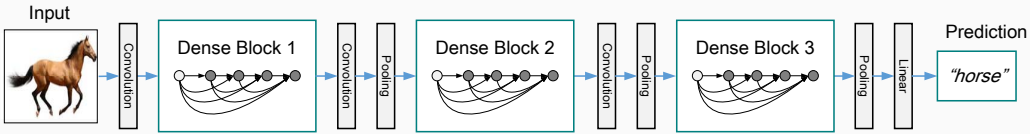
He et al, "Deep Residual Learning for Image Recognition", CVPR-2016.

Deep Residual Learning ("ResNets")



He et al, "Deep Residual Learning for Image Recognition", CVPR-2016.

Densely Connected CNNs ("DenseNets")



- Connect all layers with each other: each layer obtains additional inputs from all preceding layers and passes on its own feature-maps to all subsequent layers
- Requires $L(L+1)/2$ connections in a L-layer network
- Instead uses dense connections within "dense blocks"
- Requires fewer parameters per layer than traditional CNNs, as there is no need to re-learn redundant feature-map

Huang et al, "Densely Connected Convolutional Networks", CVPR-2017.

Summary

- Convolutional networks include local receptive fields, weight sharing, and pooling leading
- Backprop training can also be implemented as a "reverse" convolutional layer (with the weight matrix rotated)
- Implement using 4D tensors:
  - Inputs / Layer values: minibatch-size, number-fmaps, x, y
  - Weights: F_{in}, F_{out}, x, y
  - Arguments: stride, kernel size, dilation, filter groups
- Reading:

  Goodfellow et al, *Deep Learning* (ch 9)

  http://www.deeplearningbook.org/contents/convnets.html